BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Dalam tinjauan pustaka dibawah ini terdapat lima refrensi dari penelitian sebelumnya yang hampir sama dilakukan dengan penelitian yang saat ini diajukan, adapun perbandingan penelitian yang ada sebagai berikut :

Tabel 2.1 Perbandingan Metode Penelitian

Penulis	Objek	Metode	Hasil
Ananda, Laila Oktari, Silvana Rasio Henim (2014)	Citra Kata	Tesseract	Mengetahui akurasi hasil deteksi teks untuk 10 buah kata yang diterjemahkan berdasarkan jenis huruf yang digunakan dan jarak antara *smartphone* serta obyek teks
Atul Patel, Chirag Patel, Dharmendra Patel, (2012)	Citra Plat Nomor Kendaraan	Tesseract dan Trasym	Mengetahui hasil perbandingan dari Tesseract dan Trasym berdasarkan akurasi dan waktu prosesnya dari 20 citra plat nomor kendaraan
Nugroho Meganova (2015)	Citra Teks Kemasan Obat	Tesseract	Menampilkan informasi obat berdasakan teks yang telah dikenali menggunakan Tesseract dari 70 citra teks kemasan obat
Saryoto (2016)	Citra Teks	Tesseract	Menampilkan hasil deteksi teks cetak dan tulisan tangan dengan pengambilan citra di dalam dan diuar ruangan untuk studi kasus UKM Informatika dan Komputer
Aldi Setiawan (2017)	Citra Teks Bahasa Inggris	Mobile Vision	Menampilkan hasil terjemahan kata dari citra teks Bahasa Inggris ke dalam teks Bahasa Indonesia dengan dilakukan pengujian Black Box pada aplikasi yang dibangun dengan OCR Mobile Vision
Yang diusulkan (2017)	Citra Teks Bahasa Inggris	Tesseract dan Mobile Vision	Mengetahui hasil perbandingan dari Tesseract dan Mobile Vision berdasarkan waktu, ketepatan, kebutuhan memori dan resolusi citra.

Aplikasi yang diusulkan digunakan untuk membandingkan *library* Tesseract dan *library* Mobile Vision untuk pengenalan citra kata dalam Bahasa Inggris untuk citra berwarna dan citra *grayscale* yang akan dibandingkan hasil dari masing-masing *library* berdasarkan waktu, ketepatan, kebutuhan memori dan resolusi citra.

2.2. Dasar Teori

2.2.1. Pengenalan Pola

Pengenalan pola dapat dikatakan sebagai kemampuan mengenali objek berdasarkan ciri-ciri dan pengetahuan yang pernah diamatinya dari objek-objek tersebut. Tujuan dari pengenalan pola adalah mengklasifikasi dan mendeskripsikan pola atau objek kompleks melalui pengetahuan sifat-sifat atau ciri-ciri objek tersebut. (Raden Sofian Bahri dkk, 2012).

2.2.2. OCR (*Optical Character Recognition*)

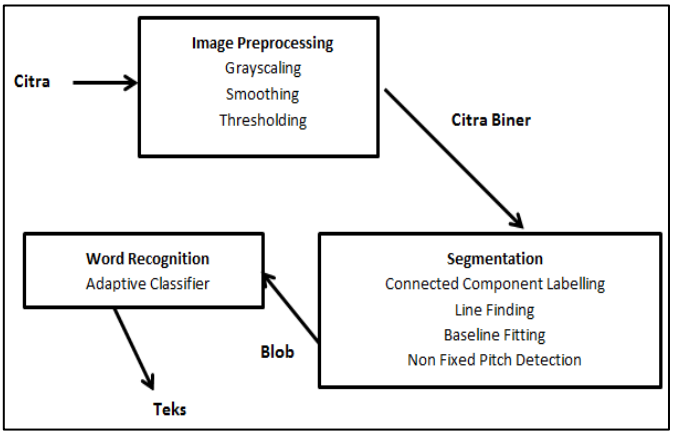
Optical Character Recognition (OCR) adalah sebuah sistem komputer yang dapat membaca huruf, baik yang berasal dari sebuah pencetak (*printer* atau mesin ketik) maupun yang berasal dari tulisan tangan. OCR adalah aplikasi yang menerjemahkan gambar karakter (*image character*) menjadi bentuk teks dengan cara menyesuaikan pola karakter per baris dengan pola yang telah tersimpan dalam *database* aplikasi. Hasil dari proses OCR adalah berupa teks sesuai dengan gambar *output scanner* dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar dan metode yang digunakan (John dkk, 2007).

Terdapat dua buah metode pada OCR, metode pertama yakni *Template Matching*. Pada dasarnya *template matching* adalah proses yang sederhana. Suatu citra masukan yang mengandung *template* tertentu dibandingkan dengan *template* pada basis data. *Template* ditempatkan pada pusat bagian citra yang akan dibandingkan dan dihitung seberapa banyak titik yang paling sesuai dengan *template*. Langkah ini diulangi terhadap keseluruhan citra masukan yang akan dibandingkan. Nilai kesesuaian titik yang paling besar antara citra masukan dan citra *template* menandakan bahwa *template* tersebut merupakan citra *template* yang paling sesuai dengan citra masukan.

Metode yang kedua dari OCR adalah *Feature Extraction*. *Feature extraction* merupakan salah satu cara untuk mengenali suatu objek dengan melihat ciri-ciri khusus yang dimiliki objek tersebut. Tujuan dari *feature extraction* adalah melakukan perhitungan dan perbandingan yang bisa digunakan untuk mengklasifikasikan ciri-ciri yang dimiliki oleh suatu citra (Raden Sofian Bahri dkk, 2012).

2.2.3. Tesseract

Tesseract merupakan *library Optical Character Recognition* (OCR) *open source* untuk berbagai sistem operasi yang awalnya dikembangkan oleh *Hewlet Packard* (HP) pada tahun 1985 hingga 1995 (Smith, 2007). Setelah tahun 2006, Tesseract dilepas oleh HP untuk digunakan secara bebas. Sejak saat itu, Tesseract dikembangkan secara luas oleh Google dan dirilis di bawah lisensi Apache 2.0 (Smith, 2007).



Gambar 2.1 Arsitektur Tesseract OCR *Engine* (Smith, 2009)

Berikut adalah penjelasan dari gambar arsitektur diatas :

1. *Image Preprocessing*

Proses *image preprocessing* terdiri dari *grayscaling, removing noise* dan *thresholding.* Bagian ini menghasilkan citra biner yang menjadi masukan pada tahap pengenalan karakter.

a. Grayscaling

Grayscaling merupakan tahapan awal dalam *image preprocessing* yang merupakan proses pengolahan citra dengan mengubah nilai *pixel* awal citra menjadi sebuah citra keabuan.

b. Smoothing

Smoothing yaitu proses pemulusan gambar untuk mengurangi *noise* dari citra.

c. Thresholding

Thresholding adalah proses untuk memisahkan antara obyek dengan *background* sehingga menghasilkan citra biner dengan mengubah gambar

berwarna maupun *grayscale* menjadi *binary image* dengan mengubah masing-masing *pixel* dalam kisaran tertentu.

2. *Segmentation*

Ada beberapa tahap dalam *segmentation* agar karakter pada citra berhasil dikenali, antara lain *connected component labelling*, *line finding*, *baseline fiting* dan *non fixed pitch detection*.

a. *Connected Component Labelling*

Connected Component Labelling adalah proses untuk mendeteksi komponen-komponen karakter yang saling terhubung, melakukan pencarian sepanjang citra kemudian mengidentifikasi *pixel* latar depan dan menandainya sebagai *outline*. *Outline* dari setiap karakter akan dikumpulkan menjadi *blob* (Smith, 2009).

b. *Line Finding*

Line Finding adalah proses untuk mencari baris teks yang dirancang supaya halaman miring dapat dikenali tanpa harus melakukan *deskew* (proses untuk mengubah halaman yang miring menjadi tegak lurus) sehingga tidak menurunkan kualitas gambar (Smith, 2009).

c. *Baseline Fitting*

Proses menentukan baris teks dan garis pangkal (*baseline*) dicocokan secara lebih tepat menggunakan *quadratic spline*. *Quadratic spline* merupakan metode untuk menghasilkan titik pada sebuah rentang data yang telah diketahui sebelumnya (Smith, 2009).

d. *Non Fixed Pitch Detection*

Bila teks yang digunakan tidak memiliki lebar garis tepi yang tetap *non fixed pitch detection* akan melakukan dengan cara mengukur batasan kesenjangan antara *base line* dengan *mean line* (Smith, 209). Baris teks dipisah ke dalam kata-kata dan dibedakan menurut jenis spasi karakter.

Tesseract menguji garis teks (*text line*) untuk menentukan apakah teks merupakan *fixed pitch*. Bila ditemukan *fixed pitch text* maka Tesseract memotong kata-kata menjadi karakter-karakter (Smith, 2009).

3. *Word Recognition*

Pengenalan selanjutnya dilakukan melalui dua proses yang disebut *pass-two* (Smith, 2007). *Pass* pertama dilakukan untuk mengenali setiap kata secara bergilir. Keberhasilan *pass* pertama dipengaruhi oleh keberadaan kata dalam kamus dan tidak mempunyai makna ganda (ambigu), sehingga kata tersebut diteruskan ke *adaptive classifier* sebagai data pelatihan. *Adaptive classifier* kemudian dapat mengenali teks lebih akurat pada halaman selanjutnya. *Pass* kedua berfungsi melakukan pengenalan kembali kata-kata yang sukar dikenali atau terlewatkan pada *pass* pertama. (Smith, 2007).

a. Pemisahan Karakter Terhubung (*Chopping Joined Characters*)

Apabila hasil dari pengenalan kata tidak memuaskan, *library* Tesseract berusaha untuk memperbaiki hasil dengan memisahkan *blob* dengan keyakinan terburuk dari pengklasifikasian karakter. Kandidat untuk titik-titik pemisahan ditemukan dari simpul cekung dari pendekatan poligonal *outline* dan mungkin saja terdapat titik cekung berlawanan lainnya atau segmen garis. Ini akan

menghabiskan sampai 3 pasang titik pemotong untuk memisahkan karakter yang terhubung dari *set ASCII* (Smith, 2009).

b. Asosiasi Karakter Patah

Ketika potongan yang potensial tidak lagi ada, ketika kata tersebut masih belum cukup baik, hal ini diberikan kepada *associator*. *Associator* membuat perencanaan A * (*Best First Search*) dari segmentasi grafik yang mungkin kombinasi dari *blob* yang dipotong secara maksimal ke dalam kandidat karakter. Ketika A * *segmentation* digunakan untuk diimplementasikan pertama kali pada tahun 1989, akurasi Tesseract terhadap karakter yang rusak cukup baik yang menjadikan Tesseract mesin komersial pada saat itu (Smith, 2009).

2.2.4. Mobile Vision

Library Mobile Vision adalah *library* yang dibuat oleh Google untuk tujuan object detection. Pada *library* Mobile Vision terdapat *Text Recognition* API, *Face* API dan *Barcode Scanner* API.

Google *developer* menjelaskan dalam websitenya, pengenalan teks adalah proses mendeteksi teks dalam gambar, video dan mengenali teks yang terkandung di dalamnya. Setelah terdeteksi, *recognizer* kemudian menentukan teks yang sebenarnya di setiap blok dan segmen ke garis dan kata-kata. Teks API mendeteksi teks dalam bahasa *berbasis* Latin (Perancis, Jerman, Inggris, dll), secara *real-time*, pada perangkat (Aldi Setiawan, 2017).

2.2.5. Sistem Operasi Android

Sistem operasi android merupakan sebuah sistem operasi berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Android

menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak (Android, 2013).

2.2.6. UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) merupakan sebuah notasi standar yang digunakan untuk melakukan pemodelan objek dunia nyata sebagai langkah pertama dalam mengembangkan metodologi desain berorientasi objek (Rizki Maulana, 2016).

Beberapa konsep dari UML yaitu :

a. UML menggunakan notasi grafis untuk menyatakan suatu desain.
b. Pemodelan dengan UML berarti menggambarkan kedalam bentuk yang dapat dipahami dengan menggunakan notasi standar UML.
c. *Unified Modeling Language* (UML) adalah bahasa standar untuk melakukan spesifikasi, visualisasi, konstruksi dan dokumentasi dari komponen-komponen perangkat lunak dan digunakan untuk pemodelan.